Group_inflation_comparison_15-22

February 11, 2025

[14]:
```
from inflation_analysis import grouping, calculate_price_indexes
from tabulate import tabulate
from tqdm import tqdm
import matplotlib.pyplot as plt
import pandas as pd


# Parameters
start_year = 2015
end_year = 2022
data_folder="/Users/roykisluk/Downloads/Consumer_Expenditure_Survey/"
top_n = 10
base_year = start_year
years=range(start_year, end_year+1)

groups, total_mmb = grouping(start_year, end_year, cex_data_folder =␣
 ↪data_folder)
groups_mmb = {key: {} for key in groups.keys()}
for key in groups:
    for year in years:
        groups_mmb[key][year] = groups[key][year][['misparmb']]

group_analysis = {}
for key in groups.keys():
    group_number = list(groups.keys()).index(key) + 1
    total_groups = len(groups)
    print(f"Group {group_number}/{total_groups} ({key}) started.")
    combined_df, combined_secondary_df, combined_primary_df, yearly_price_index␣
 ↪= calculate_price_indexes(
        start_year, end_year, base_year, group_mmb=groups_mmb[key],␣
 ↪cex_data_folder=data_folder, verbose=False
    )
    group_analysis[key] = {
        'combined_secondary_df': combined_secondary_df,
        'combined_primary_df': combined_primary_df,
        'yearly_price_index': yearly_price_index
    }
```

```
    print(f"Group {group_number}/{total_groups} ({key}) successfully computed.")
```

```
Group 1/11 (Arabs) started.

Loading price data: 100%|      | 8/8 [00:08<00:00,  1.01s/it]
Calculating price indexes: 100%|      | 8/8 [00:08<00:00,  1.08s/it]

Group 1/11 (Arabs) successfully computed.
Group 2/11 (Haredi) started.

Loading price data: 100%|      | 8/8 [00:06<00:00,  1.17it/s]
Calculating price indexes: 100%|      | 8/8 [00:07<00:00,  1.07it/s]

Group 2/11 (Haredi) successfully computed.
Group 3/11 (Low_inc) started.

Loading price data: 100%|      | 8/8 [00:06<00:00,  1.18it/s]
Calculating price indexes: 100%|      | 8/8 [00:07<00:00,  1.11it/s]

Group 3/11 (Low_inc) successfully computed.
Group 4/11 (High_inc) started.

Loading price data: 100%|      | 8/8 [00:06<00:00,  1.17it/s]
Calculating price indexes: 100%|      | 8/8 [00:11<00:00,  1.46s/it]

Group 4/11 (High_inc) successfully computed.
Group 5/11 (Young) started.

Loading price data: 100%|      | 8/8 [00:06<00:00,  1.18it/s]
Calculating price indexes: 100%|      | 8/8 [00:09<00:00,  1.19s/it]

Group 5/11 (Young) successfully computed.
Group 6/11 (Old) started.

Loading price data: 100%|      | 8/8 [00:07<00:00,  1.13it/s]
Calculating price indexes: 100%|      | 8/8 [00:11<00:00,  1.42s/it]

Group 6/11 (Old) successfully computed.
Group 7/11 (Low_SES) started.

Loading price data: 100%|      | 8/8 [00:08<00:00,  1.06s/it]
Calculating price indexes: 100%|      | 8/8 [00:07<00:00,  1.09it/s]

Group 7/11 (Low_SES) successfully computed.
Group 8/11 (High_SES) started.

Loading price data: 100%|      | 8/8 [00:08<00:00,  1.02s/it]
Calculating price indexes: 100%|      | 8/8 [00:04<00:00,  1.73it/s]

Group 8/11 (High_SES) successfully computed.
Group 9/11 (Muslim) started.

Loading price data: 100%|      | 8/8 [00:08<00:00,  1.10s/it]
Calculating price indexes: 100%|      | 8/8 [00:07<00:00,  1.06it/s]

Group 9/11 (Muslim) successfully computed.
Group 10/11 (Christian) started.
```

```
Loading price data: 100%|      | 8/8 [00:07<00:00,  1.01it/s]
Calculating price indexes: 100%|      | 8/8 [00:04<00:00,  1.67it/s]

Group 10/11 (Christian) successfully computed.
Group 11/11 (Druze) started.

Loading price data: 100%|      | 8/8 [00:08<00:00,  1.03s/it]
Calculating price indexes: 100%|      | 8/8 [00:03<00:00,  2.19it/s]

Group 11/11 (Druze) successfully computed.
```

[2]:
```
gen_pop_df, gen_pop_secondary_df, gen_pop_primary_df,␣
 ↪gen_pop_yearly_price_index = calculate_price_indexes(start_year, end_year,␣
 ↪base_year, cex_data_folder=data_folder, verbose=False)
```

```
Loading price data: 100%|      | 8/8 [00:08<00:00,  1.00s/it]
Calculating price indexes: 100%|      | 8/8 [00:33<00:00,  4.14s/it]
```

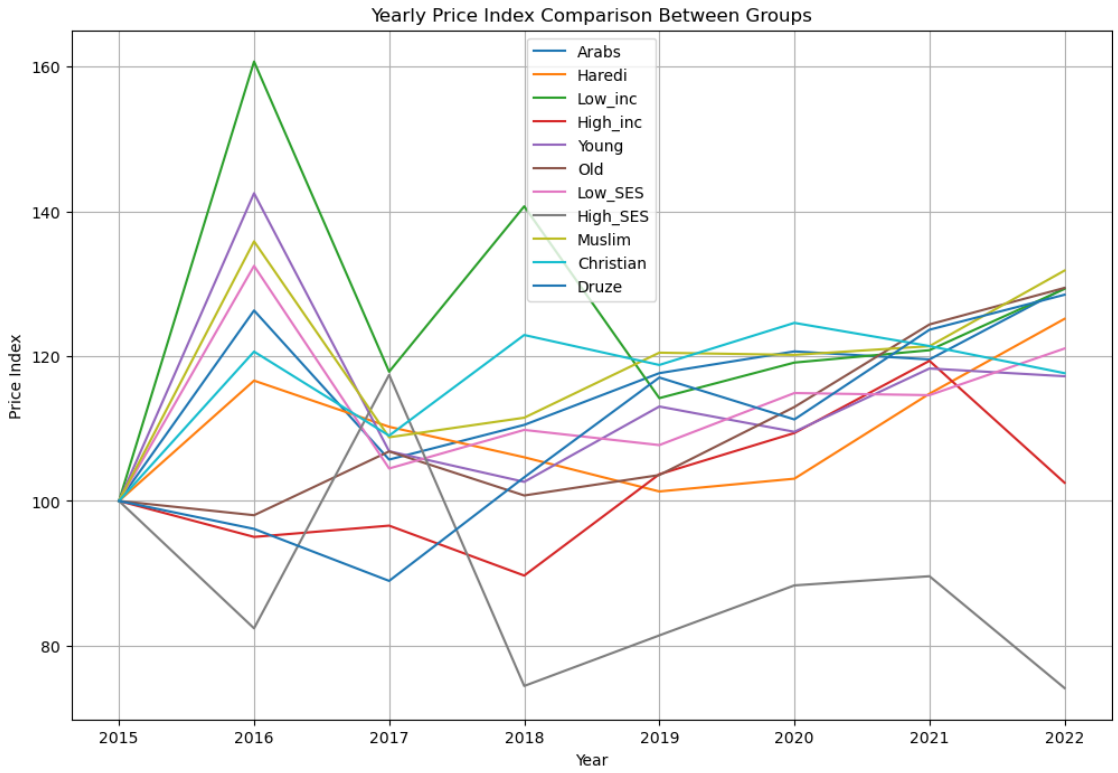
[3]:
```
group_counts = {group: {year: len(groups_mmb[group][year]) for year in␣
 ↪groups_mmb[group]} for group in groups_mmb}
# Create a dataframe with number of observations per year per group
observations_df = pd.DataFrame(group_counts).T

# Calculate the relative share of each group per year
total_observations_per_year = observations_df.sum(axis=0)
relative_share_df = observations_df.div(total_observations_per_year, axis=1) *␣
 ↪100

# Combine the absolute and relative values into a single dataframe
combined_df = observations_df.join(relative_share_df, rsuffix='_share')

# Display the dataframe
print(tabulate(combined_df, headers='keys', tablefmt='psql'))
```

```
+-----------+--------+--------+--------+--------+--------+--------+--------+----
----+-------------+-------------+-------------+-------------+-------------
+-------------+-------------+-------------+
|           |   2015 |   2016 |   2017 |   2018 |   2019 |   2020 |   2021 |
2022 |   2015_share |   2016_share |   2017_share |   2018_share |   2019_share
|   2020_share |   2021_share |   2022_share |
|-----------+--------+--------+--------+--------+--------+--------+--------+----
----+-------------+-------------+-------------+-------------+-------------
+-------------+-------------+-------------|
| Arabs     |   1136 |   1273 |   1327 |   1145 |   1103 |    513 |    951 |
727 |     11.5236  |    11.9351   |    12.2825   |     11.1295  |     10.9273  |
7.49014  |     11.1476  |      9.51944 |
| Haredi    |    734 |    778 |    757 |    786 |    565 |    440 |    551 |
595 |      7.44573 |     7.29421  |     7.00666  |      7.63997 |      5.59738 |
```

```
6.4243   |      6.4588  |      7.79102 |
| Low_inc    |    873 |    890 |    940 |    880 |    646 |    397 |    503 |
515 |      8.85575 |     8.34427  |     8.70048  |      8.55365 |      6.39984 |
5.79647  |      5.89614 |      6.74349 |
| High_inc   |   1667 |   1770 |   1741 |   1766 |   1848 |   1432 |   1479 |
1162 |     16.9101  |    16.5948   |    16.1144   |     17.1656  |     18.3079
|    20.9082   |     17.3368  |     15.2154  |
| Young      |   1340 |   1354 |   1358 |   1278 |   1108 |    718 |    877 |
820 |     13.593   |    12.6945   |    12.5694   |     12.4222  |     10.9768  |
10.4833   |     10.2802  |     10.7372  |
| Old        |   2091 |   2372 |   2348 |   2375 |   2279 |   1786 |   1779 |
1663 |     21.2112  |    22.2389   |    21.7327   |     23.0851  |     22.5778
|    26.0768   |     20.8534  |     21.7756  |
| Low_SES    |    616 |    695 |    728 |    654 |   1146 |    807 |   1204 |
1211 |      6.24873 |     6.51603  |     6.73825  |      6.35692 |     11.3533
|    11.7827   |     14.1132  |     15.857   |
| High_SES   |    121 |     95 |    102 |    127 |    189 |    163 |    153 |
146 |      1.22743 |     0.890681 |     0.944095 |      1.23445 |      1.8724  |
2.37991  |      1.79346 |      1.91175 |
| Muslim     |    880 |    999 |   1056 |    920 |    870 |    381 |    710 |
583 |      8.92676 |     9.36621  |     9.77416  |      8.94246 |      8.61898 |
5.56286  |      8.32259 |      7.63389 |
| Christian |    260 |    309 |    300 |    234 |    231 |    151 |    205 |
137 |      2.63745 |     2.89706  |     2.77675  |      2.27449 |      2.28849 |
2.2047   |      2.403   |      1.7939  |
| Druze      |    140 |    131 |    147 |    123 |    109 |     61 |    119 |
78 |      1.42017 |     1.2282   |     1.36061  |      1.19557 |      1.07985 |
0.890641 |      1.39491 |      1.02134 |
+-----------+--------+--------+--------+--------+--------+--------+--------+----
----+--------------+--------------+--------------+--------------+-------------
+--------------+--------------+--------------+
```

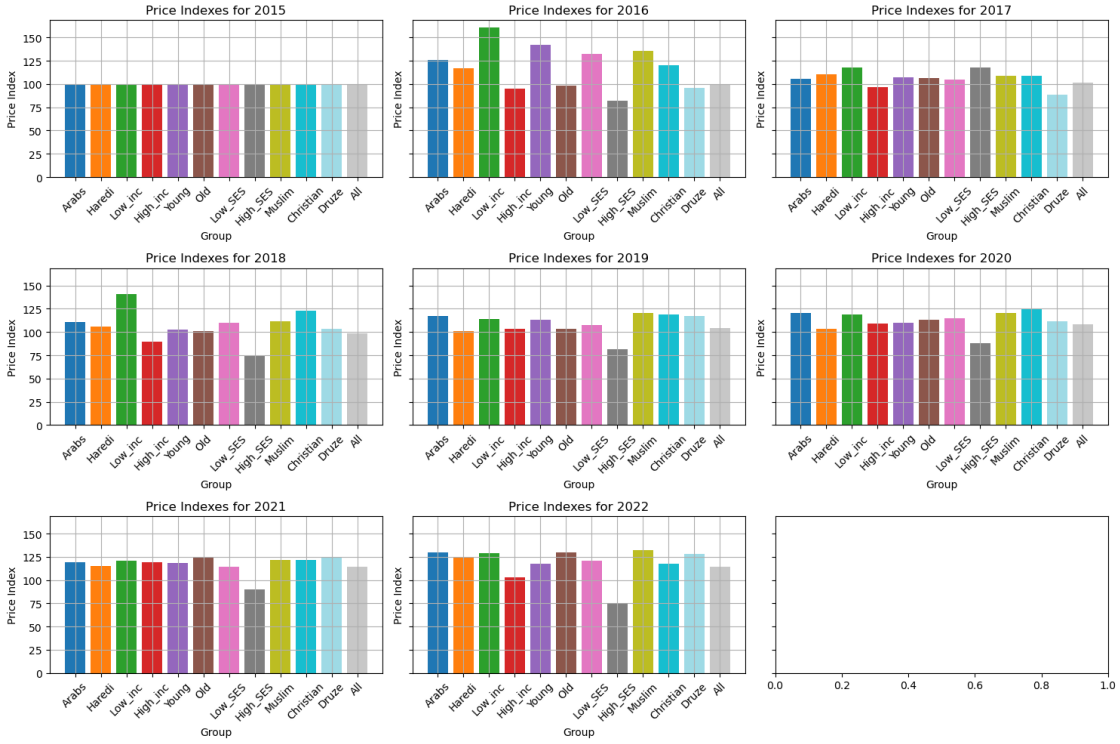
[4]:
```python
# Extract yearly price indexes for each group
group_yearly_price_indexes = {group:␣
 ↪group_analysis[group]['yearly_price_index'] for group in group_analysis}

# Plot the yearly price indexes
plt.figure(figsize=(12, 8))
for group, price_indexes in group_yearly_price_indexes.items():
    years = list(price_indexes.keys())
    indexes = list(price_indexes.values())
    plt.plot(years, indexes, label=group)

plt.xlabel('Year')
plt.ylabel('Price Index')
plt.title('Yearly Price Index Comparison Between Groups')
plt.legend()
```

```
plt.grid(True)
plt.show()
```



[17]:
```
# Extract yearly price indexes for each group including general population
group_yearly_price_indexes = {group:␣
 ↪group_analysis[group]['yearly_price_index'] for group in group_analysis}
group_yearly_price_indexes['All'] = gen_pop_yearly_price_index

# Define colors for each group
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b',␣
 ↪'#e377c2', '#7f7f7f', '#bcbd22', '#17becf', '#9edae5', '#c7c7c7']

# Create subplots for each year
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(15, 10), sharey=True)
axes = axes.flatten()

for i, year in enumerate(years[:len(axes)]):
    ax = axes[i]
    groups = list(group_yearly_price_indexes.keys())
    price_indexes = [group_yearly_price_indexes[group][year] for group in␣
 ↪groups]
    ax.bar(groups, price_indexes, color=colors)
```

```
    ax.set_title(f'Price Indexes for {year}')
    ax.set_xlabel('Group')
    ax.set_ylabel('Price Index')
    ax.grid(True)
    ax.tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.show()
```



[18]:
```
fig, axes = plt.subplots(nrows=4, ncols=3, figsize=(20, 15), sharey=True)
axes = axes.flatten()

for i, (group, data) in enumerate(group_analysis.items()):
    yearly_price_index = data['yearly_price_index']
    years = list(yearly_price_index.keys())
    indexes = list(yearly_price_index.values())

    axes[i].bar(years, indexes, color='skyblue')
    axes[i].set_title(group)
    axes[i].set_xlabel('Year')
    axes[i].set_ylabel('Price Index')
    axes[i].grid(True)
```

```
# Remove any empty subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```

[19]:
```
for group, analysis in group_analysis.items():
    # Calculate the weight differences between the group and the general␣
 ↪population for secondary categories
    weight_diff_df = analysis['combined_secondary_df'].copy()
    weight_diff_df['weight_diff'] = weight_diff_df['weight'] -␣
 ↪gen_pop_secondary_df['weight']

    # Sort by the absolute value of the weight differences in descending order
    weight_diff_df['abs_weight_diff'] = weight_diff_df['weight_diff'].abs()
    sorted_weight_diff_df = weight_diff_df.sort_values(by='abs_weight_diff',␣
 ↪ascending=False)

    # Display the top_n largest gaps using tabulate
```

```
    print(f"Group: {group}")
    print(tabulate(sorted_weight_diff_df.head(top_n)[['Year', 'prodcode',␣
↪'description', 'weight', 'weight_diff']], headers='keys', tablefmt='psql'))
    print("\n")
```

```
Group: Arabs
+-----+--------+------------+----------------------------------+------------+---
------------+
|     |   Year |   prodcode | description                      |     weight |
weight_diff |
|-----+--------+------------+----------------------------------+------------+---
------------|
| 378 |   2020 |        395 | Personal Jewelry And Watches     | 0.00219264 |
-0.135039 |
| 314 |   2019 |        394 | Legal And Other Services         | 0.00540572 |
-0.130674 |
| 249 |   2018 |        392 | Personal Services And Cosmetics  | 0.00436179 |
-0.129883 |
| 119 |   2016 |        384 | Other Transportation Expenses    | 0.00644553 |
-0.126804 |
| 371 |   2020 |        383 | Vehicle Expenses                 | 0.127396   |
0.11329  |
| 184 |   2017 |        385 | Mail Telephone And Communication | 0.0216433  |
-0.112431 |
| 497 |   2022 |        383 | Vehicle Expenses                 | 0.122325   |
0.109744 |
| 308 |   2019 |        383 | Vehicle Expenses                 | 0.121403   |
0.105517 |
| 118 |   2016 |        383 | Vehicle Expenses                 | 0.118808   |
0.105059 |
| 182 |   2017 |        383 | Vehicle Expenses                 | 0.117387   |
0.103783 |
+-----+--------+------------+----------------------------------+------------+---
------------+


Group: Haredi
+-----+--------+------------+------------------------------------------+---------
---+---------------+
|     |   Year |   prodcode | description                              |
weight |   weight_diff |
|-----+--------+------------+------------------------------------------+---------
---+---------------|
| 314 |   2019 |        394 | Legal And Other Services                 |
0.0045246  |    -0.131556  |
| 508 |   2022 |        398 | Expenses Not Elsewhere Specified         |
0.00594857 |    -0.130851  |
|  54 |   2015 |        384 | Other Transportation Expenses            |
```

```
0.00228819 |    -0.130432  |
| 184 |   2017 |        391 | Cigarettes Tobacco And Smoking Supplies |
0.00470577 |    -0.129368  |
| 249 |   2018 |        392 | Personal Services And Cosmetics         |
0.00564004 |    -0.128604  |
| 119 |   2016 |        385 | Mail Telephone And Communication        |
0.00672907 |    -0.126521  |
| 378 |   2020 |        396 | Bags Suitcases And Other Products       |
0.0155506  |    -0.121681  |
| 443 |   2021 |        397 | Organization Fees And Donations         |
0.0130076  |    -0.0887514 |
| 256 |   2019 |        300 | Bread Grains and Pastries               |
0.0745475  |     0.0693823 |
| 194 |   2018 |        302 | Meat And Poultry                        | 0.072473
|     0.0689943 |
+-----+--------+------------+-----------------------------------------+---------
---+---------------+
```

```
Group: Low_inc
+-----+--------+------------+-----------------------------------------+---------
---+---------------+
|     |   Year |   prodcode | description                             |
weight |   weight_diff |
|-----+--------+------------+-----------------------------------------+---------
---+---------------|
| 314 |   2019 |        394 | Legal And Other Services                |
0.00355971 |    -0.13252   |
| 378 |   2020 |        396 | Bags Suitcases And Other Products       |
0.0135375  |    -0.123694  |
| 184 |   2017 |        385 | Mail Telephone And Communication        |
0.0140138  |    -0.12006   |
| 249 |   2018 |        391 | Cigarettes Tobacco And Smoking Supplies |
0.0267694  |    -0.107475  |
| 443 |   2021 |        398 | Expenses Not Elsewhere Specified        |
0.00232739 |    -0.0994316 |
| 321 |   2020 |        302 | Meat And Poultry                        |
0.0841995  |     0.078965  |
| 259 |   2019 |        302 | Meat And Poultry                        |
0.0815777  |     0.0780907 |
| 370 |   2020 |        383 | Vehicle Expenses                        |
0.0740601  |     0.0706119 |
| 446 |   2022 |        302 | Meat And Poultry                        | 0.083766
|     0.0675385 |
| 246 |   2018 |        383 | Vehicle Expenses                        |
0.0694858  |     0.0644607 |
+-----+--------+------------+-----------------------------------------+---------
---+---------------+
```

Group: High_inc

	Year	prodcode	description	weight	weight_diff
505	2022	383	Vehicle Expenses	0.201978	0.19684
312	2019	383	Vehicle Expenses	0.19702	0.1832
248	2018	383	Vehicle Expenses	0.195316	0.181458
378	2020	385	Mail Telephone And Communication	0.00757866	-0.129653
249	2018	384	Other Transportation Expenses	0.00461579	-0.129629
314	2019	385	Mail Telephone And Communication	0.00675151	-0.129329
508	2022	391	Cigarettes Tobacco And Smoking Supplies	0.00757757	-0.129222
443	2021	385	Mail Telephone And Communication	0.00761718	-0.0941419
376	2020	383	Vehicle Expenses	0.106518	0.0925092
441	2021	383	Vehicle Expenses	0.105877	0.0917627

Group: Young

	Year	prodcode	description	weight	weight_diff
508	2022	392	Personal Services And Cosmetics	0.00873514	-0.128065
184	2017	384	Other Transportation Expenses	0.0082545	-0.12582
311	2019	383	Vehicle Expenses	0.128955	0.123859
249	2018	385	Mail Telephone And Communication	0.0168145	-0.11743

314	2019	391	Cigarettes Tobacco And Smoking Supplies	0.023755	-0.112325
378	2020	391	Cigarettes Tobacco And Smoking Supplies	0.0261559	-0.111075
375	2020	383	Vehicle Expenses	0.115443	0.110277
247	2018	383	Vehicle Expenses	0.108966	0.0953299
183	2017	383	Vehicle Expenses	0.108081	0.0942552
439	2021	383	Vehicle Expenses	0.113704	0.0929

Group: Old

	Year	prodcode	description	weight	weight_diff
119	2016	384	Other Transportation Expenses	0.0035588	-0.129691
378	2020	391	Cigarettes Tobacco And Smoking Supplies	0.0109076	-0.126324
508	2022	392	Personal Services And Cosmetics	0.0110679	-0.125732
314	2019	385	Mail Telephone And Communication	0.0107218	-0.125358
249	2018	385	Mail Telephone And Communication	0.0105258	-0.123719
184	2017	385	Mail Telephone And Communication	0.0109945	-0.12308
312	2019	383	Vehicle Expenses	0.13332	0.1195
118	2016	383	Vehicle Expenses	0.130615	0.116866
247	2018	383	Vehicle Expenses	0.130022	0.116385
504	2022	383	Vehicle Expenses	0.134459	0.11442

Group: Low_SES

	Year	prodcode	description	weight	weight_diff
508	2022	397	Organization Fees And Donations	0.00208536	-0.134715
314	2019	392	Personal Services And Cosmetics	0.00498447	-0.131096
119	2016	384	Other Transportation Expenses	0.00683727	-0.126413
378	2020	393	Personal Products And Cosmetics	0.0198475	-0.117384
184	2017	385	Mail Telephone And Communication	0.0198135	-0.114261
249	2018	391	Cigarettes Tobacco And Smoking Supplies	0.0289928	-0.105252
373	2020	383	Vehicle Expenses	0.110699	0.103661
131	2017	302	Meat And Poultry	0.107198	0.100326
246	2018	383	Vehicle Expenses	0.105049	0.100024
443	2021	395	Personal Jewelry And Watches	0.002078	-0.099681

Group: High_SES

	Year	prodcode	description	weight	weight_diff
166	2017	362	Dental Care	0.210163	0.208672
349	2020	362	Dental Care	0.211731	0.198522
469	2022	362	Dental Care	0.198507	0.198149
290	2019	362	Dental Care	0.195425	0.190394
408	2021	362	Dental Care	0.203472	0.181465
104	2016	362	Dental Care	0.193576	

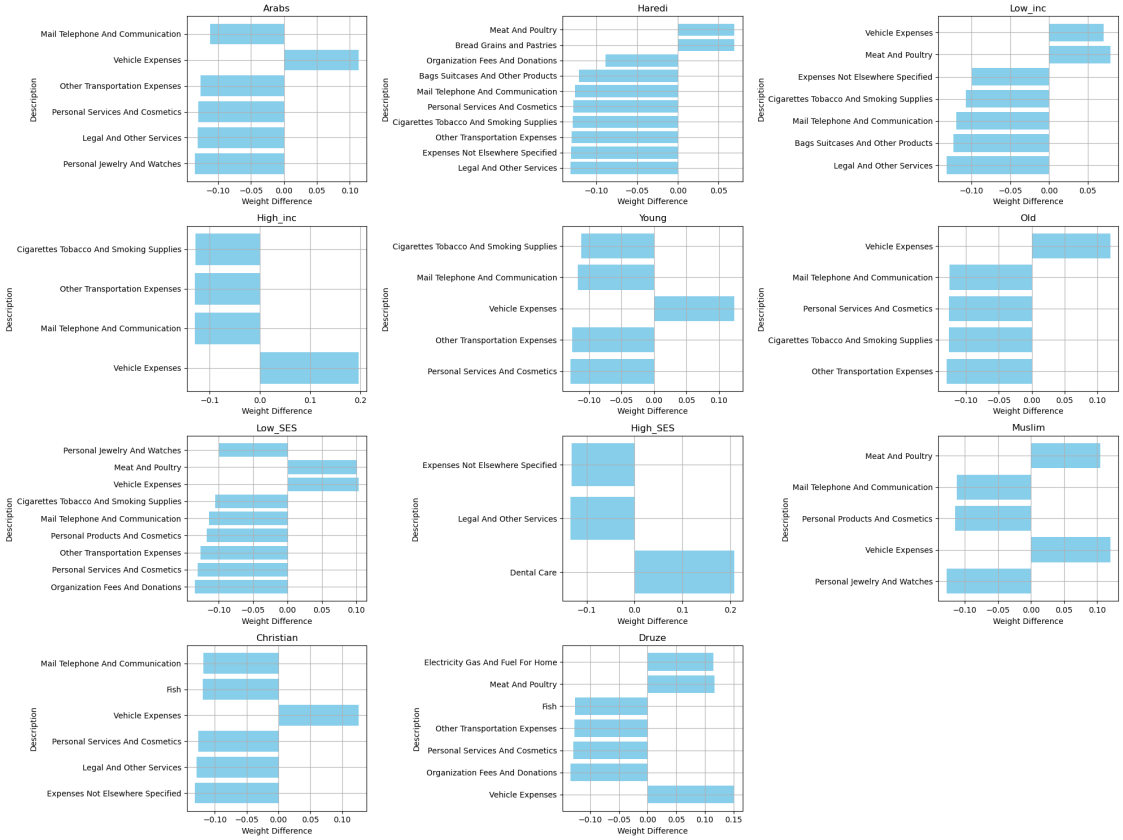
```
0.178436 |
| 227 |   2018 |        362 | Dental Care                       | 0.182658    |
0.171991 |
|  41 |   2015 |        362 | Dental Care                       | 0.143745    |
0.140754 |
| 184 |   2017 |        394 | Legal And Other Services          | 0.000596769 |
-0.133477 |
| 249 |   2018 |        398 | Expenses Not Elsewhere Specified | 0.0023647   |
-0.13188  |
+-----+--------+------------+-----------------------------------+-------------+--
-------------+


Group: Muslim
+-----+--------+------------+-----------------------------------+------------+---
------------+
|     |   Year |   prodcode | description                       |     weight |
weight_diff |
|-----+--------+------------+-----------------------------------+------------+---
------------|
| 314 |   2019 |        395 | Personal Jewelry And Watches      | 0.00843613 |
-0.127644 |
| 492 |   2022 |        383 | Vehicle Expenses                  | 0.120406   |
0.119781 |
| 430 |   2021 |        383 | Vehicle Expenses                  | 0.117571   |
0.114532 |
| 249 |   2018 |        393 | Personal Products And Cosmetics   | 0.0200521  |
-0.114192 |
| 119 |   2016 |        385 | Mail Telephone And Communication | 0.0209635  |
-0.112286 |
| 367 |   2020 |        383 | Vehicle Expenses                  | 0.122417   |
0.109624 |
| 181 |   2017 |        383 | Vehicle Expenses                  | 0.113309   |
0.108097 |
| 244 |   2018 |        383 | Vehicle Expenses                  | 0.113311   |
0.105903 |
| 194 |   2018 |        302 | Meat And Poultry                  | 0.108024   |
0.104545 |
| 117 |   2016 |        383 | Vehicle Expenses                  | 0.115908   |
0.10238  |
+-----+--------+------------+-----------------------------------+------------+---
------------+


Group: Christian
+-----+--------+------------+-----------------------------------+------------+---
------------+
|     |   Year |   prodcode | description                       |     weight |
```

```
weight_diff |
|-----+--------+------------+----------------------------------+------------+---
------------|
| 249 |   2018 |        398 | Expenses Not Elsewhere Specified | 0.00292803 |
-0.131316 |
| 184 |   2017 |        394 | Legal And Other Services         | 0.00567489 |
-0.128399 |
| 119 |   2016 |        392 | Personal Services And Cosmetics  | 0.00716718 |
-0.126083 |
| 300 |   2019 |        383 | Vehicle Expenses                 | 0.127235   |
0.125994 |
|  52 |   2015 |        383 | Vehicle Expenses                 | 0.135027   |
0.121572 |
| 478 |   2022 |        383 | Vehicle Expenses                 | 0.130368   |
0.119949 |
| 314 |   2020 |        303 | Fish                             | 0.0173376  |
-0.118743 |
|  54 |   2015 |        385 | Mail Telephone And Communication | 0.0145448  |
-0.118176 |
| 178 |   2017 |        383 | Vehicle Expenses                 | 0.127629   |
0.11199  |
| 417 |   2021 |        383 | Vehicle Expenses                 | 0.130671   |
0.109771 |
+-----+--------+------------+----------------------------------+------------+---
------------+
```

```
Group: Druze
+-----+--------+------------+----------------------------------+-------------+-
--------------+
|     |   Year |   prodcode | description                      |      weight |
weight_diff |
|-----+--------+------------+----------------------------------+-------------+-
--------------|
| 405 |   2021 |        383 | Vehicle Expenses                 | 0.15883     |
0.15128  |
| 236 |   2018 |        383 | Vehicle Expenses                 | 0.140262    |
0.137399 |
| 184 |   2017 |        397 | Organization Fees And Donations  | 0.000349413 |
-0.133725 |
| 119 |   2016 |        392 | Personal Services And Cosmetics  | 0.00436819  |
-0.128882 |
|  54 |   2015 |        384 | Other Transportation Expenses    | 0.00503555  |
-0.127685 |
| 249 |   2019 |        303 | Fish                             | 0.0084111   |
-0.125833 |
| 348 |   2020 |        383 | Vehicle Expenses                 | 0.135245    |
0.124897 |
```

```
| 462 |   2022 |        383 | Vehicle Expenses                   | 0.167994    |
0.119967 |
| 361 |   2021 |        302 | Meat And Poultry                   | 0.117745    |
0.116868 |
| 322 |   2020 |        332 | Electricity Gas And Fuel For Home | 0.122888    |
0.115144 |
+-----+--------+------------+------------------------------------+-------------+-
--------------+
```

[20]:
```
fig, axes = plt.subplots(nrows=4, ncols=3, figsize=(20, 15), sharey=False)
axes = axes.flatten()

for i, (group, analysis) in enumerate(group_analysis.items()):
    # Calculate the weight differences between the group and the general␣
 ↪population for secondary categories
    weight_diff_df = analysis['combined_secondary_df'].copy()
    weight_diff_df['weight_diff'] = weight_diff_df['weight'] -␣
 ↪gen_pop_secondary_df['weight']

    # Sort by the absolute value of the weight differences in descending order
    weight_diff_df['abs_weight_diff'] = weight_diff_df['weight_diff'].abs()
    sorted_weight_diff_df = weight_diff_df.sort_values(by='abs_weight_diff',␣
 ↪ascending=False)

    # Replace NaN values in 'description' with an empty string
    sorted_weight_diff_df['description'] = sorted_weight_diff_df['description'].
 ↪fillna('')

    # Select the top n largest gaps
    top_n_weight_diff_df = sorted_weight_diff_df.head(top_n)

    # Plot the top n largest gaps
    axes[i].barh(top_n_weight_diff_df['description'],␣
 ↪top_n_weight_diff_df['weight_diff'], color='skyblue')
    axes[i].set_title(group)
    axes[i].set_xlabel('Weight Difference')
    axes[i].set_ylabel('Description')
    axes[i].grid(True)

# Remove any empty subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```



[21]:
```
fig, axes = plt.subplots(nrows=4, ncols=3, figsize=(20, 15), sharey=False)
axes = axes.flatten()

for i, (group, analysis) in enumerate(group_analysis.items()):
    secondary_df = analysis['combined_secondary_df']
    # Filter for increases in price indexes
    increased_price_df = secondary_df[secondary_df['price_index'] > 100]
    # Calculate the contribution to the yearly price index
    increased_price_df = increased_price_df.copy()  # Ensure you're working␣
↪with a copy
    increased_price_df.loc[:, 'contribution'] =␣
↪increased_price_df['price_index'] * increased_price_df['weight']
    # Sort by contribution in descending order
    top_contributors = increased_price_df.sort_values(by='contribution',␣
↪ascending=False).head(top_n)

    # Replace NaN values in 'description' with a placeholder text
    top_contributors['description'] = top_contributors['description'].
↪fillna('No Description')
```

```
    # Plot the top contributors
    axes[i].barh(top_contributors['description'],␣
 ↪top_contributors['contribution'], color='skyblue')
    axes[i].set_title(group)
    axes[i].set_xlabel('Contribution to Price Index')
    axes[i].set_ylabel('Description')
    axes[i].grid(True)

# Remove any empty subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```